# C# Design Patterns: Visitor

## IMPLEMENTING THE VISITOR PATTERN

Harrison Ferrone
SOFTWARE DEVELOPER

@journeyman_programmer   www.paradigmshiftdev.com

# Overview

**The Visitor Pattern in Practice:**

- Defining the visitor blueprint
- Creating concrete visitor classes
- Using an object structure
- Reviewing use cases/practical applications

# Design Pattern Categories

Creational

Structural

Behavioral

“Represent an operation to be performed on the elements of an object structure without changing the classes of the elements on which it operates.

**Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides,** *Design Patterns: Elements of Reusable Object-Oriented Software*

**Behavior can be added to existing hierarchy**

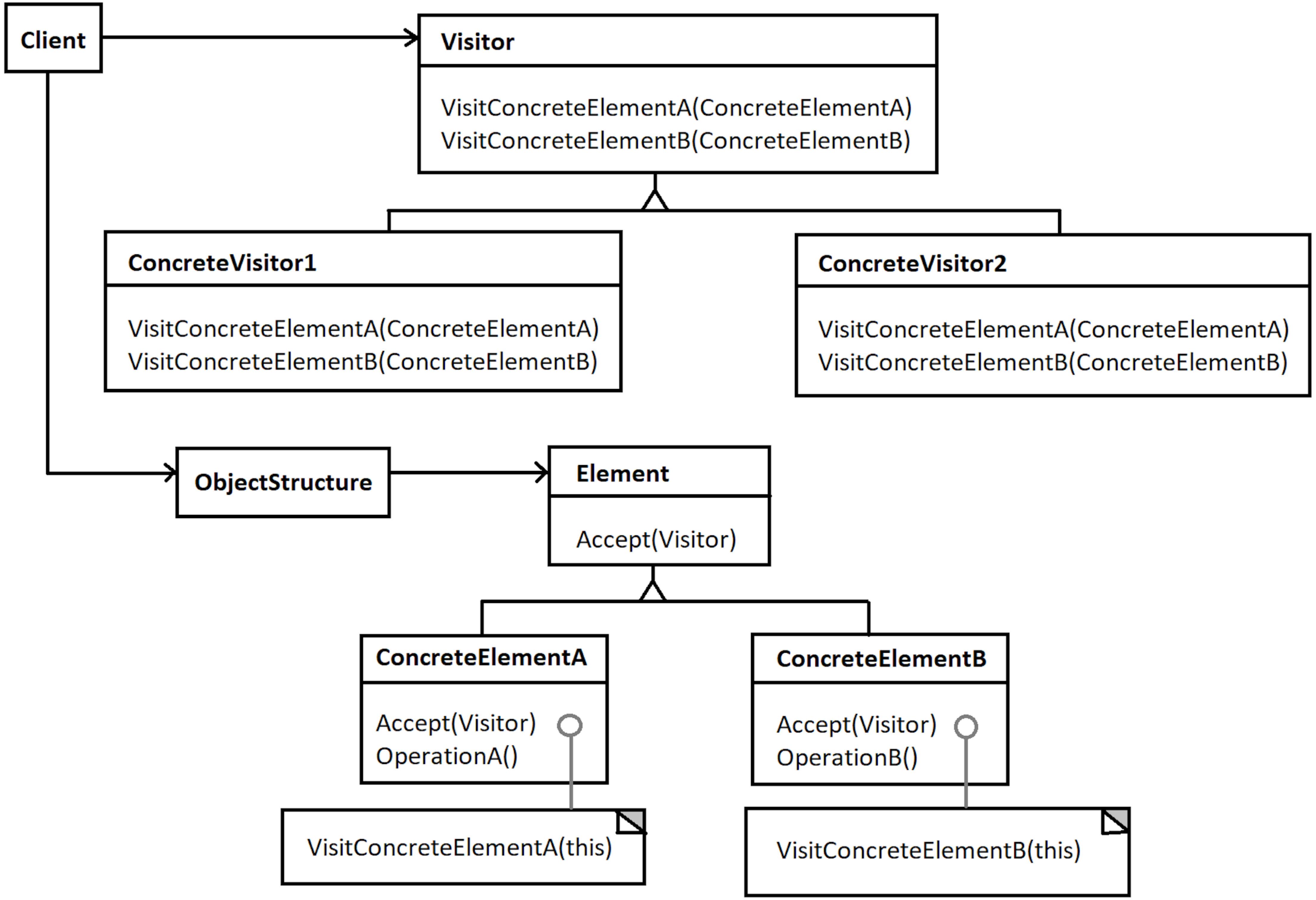
- No changes to underlying classes

**Behavior is class-specific**

**No hierarchical connection is necessary**

- As long as classes are marked as visit-able

# Visitor Pattern Diagram



Client
Visitor
VisitConcreteElementA(ConcreteElementA)
VisitConcreteElementB(ConcreteElementB)
ConcreteVisitor1
VisitConcreteElementA(ConcreteElementA)
VisitConcreteElementB(ConcreteElementB)
ConcreteVisitor2
VisitConcreteElementA(ConcreteElementA)
VisitConcreteElementB(ConcreteElementB)
ObjectStructure
Element
Accept(Visitor)
ConcreteElementA
Accept(Visitor)
OperationA()
ConcreteElementB
Accept(Visitor)
OperationB()
VisitConcreteElementA(this)
VisitConcreteElementB(this)

When a project has several different classes, with different interfaces, that need additional behavior without changing their underlying structure

# Creating the Visitor Blueprint

# Implementing a Concrete Visitor

# Testing the Visitor Pattern

# Adding an Object Structure

# Use Cases and Applications

# Visitor Use Cases

Not for everything – all added class behavior doesn’t need to be implemented with the visitor pattern

When a project has a variety of classes, with different interfaces and inheritance structures, that need added class-specific behavior

When different and unrelated behaviors need to be applied without polluting the existing classes

When the existing class structure is unlikely to change, but new behavior needs to be added at will

# Design Pattern Implications

Adding behavior across a wide range of differing exiting classes

Gathering related behaviors and accumulated state across differing classes

Changing the class hierarchy while using the Visitor pattern can be costly – you'll have to update the interface for all visitors to match

Encapsulation can potentially be broken, as the visitor pattern often needs access to the internal state of your elements to do its job

# Related Patterns

Composite

Interpreter

# Summary

How to create the visitor and element interfaces

How to create concrete visitor and element classes

How to work with an object structure

How to identify suitable use cases and potential drawbacks

Leave a course rating

Use the Discussion section as a resource

Follow my author profile for updates

Get in touch and let’s talk code!

Happy Coding!